Soft Clustering for Nonparametric Probability Density Function Estimation

Ezequiel López-Rubio, Juan Miguel Ortiz-de-Lazcano-Lobato,
Domingo López-Rodríguez, and María del Carmen Vargas-González

School of Computer Engineering
University of Málaga
Campus de Teatinos, s/n. 29071
Málaga Spain
Phone: (+34) 95 213 71 55
Fax: (+34) 95 213 13 97
{ezeqlr,jmortiz}@lcc.uma.es, dlopez@ctima.uma.es

Abstract. We present a nonparametric probability density estimation model. The classical Parzen window approach builds a spherical Gaussian density around every input sample. Our method has a first stage where hard neighbourhoods are determined for every sample. Then soft clusters are considered to merge the information coming from several hard neighbourhoods. Our proposal estimates the local principal directions to yield a specific Gaussian mixture component for each soft cluster. This leads to outperform other proposals where local parameter selection is not allowed and/or there are no smoothing strategies, like the manifold Parzen windows.

Keywords: Probability density estimation, nonparametric modeling, soft clustering, Parzen windows.

1 Introduction

The estimation of the unknown probability density function (PDF) of a continuous distribution from a set of data points forming a representative sample drawn from the underlying density is a problem of fundamental importance to all aspects of machine learning and pattern recognition (see [1], [2] and [3]).

Parametric approaches make assumptions about the unknown distribution. They consider, a priori, a particular functional form for the PDF and reduce the problem to the estimation of the required functional parameters. On the other hand, nonparametric methods make less rigid assumptions. Popular nonparametric methods include the histogram, kernel estimation, nearest neighbor methods and restricted maximum likelihood methods, as can be found in [4], [5], [6] and [7].

The kernel density estimator, also commonly referred as the Parzen window estimator, [9], places a Gaussian kernel on each data point of the training set. Then, the PDF is approximated by summing all the kernels, which are multiplied by a normalizing factor. Thus, this model can be viewed as a finite mixture model (see [8]) where the number of mixture components will equal the number of points in the data

J. Marques de Sá et al. (Eds.): ICANN 2007, Part I, LNCS 4668, pp. 707–716, 2007.

sample. Parzen windows estimates are usually built using a 'spherical Gaussian' with a single scalar variance parameter σ^2, which spreads the density mass equally along all input space directions and gives too much probability to irrelevant regions of space and too little along the principal directions of variance of the distribution. This drawback is partially solved in Manifold Parzen Windows algorithm [10], where a different covariance matrix is calculated for each component. The covariance matrix is estimated by considering a hard neighbourhood of each input sample. We propose in Section 2 to build soft clusters to share the information among neighbourhoods. This leads to filter the input noise by smoothing the estimated parameters.

We present in section 3 a method that automatizes the selection of the right dimensionality of the manifold. The asymptotical convergence of the proposed method is formally proven in Section 4. We show some experimental results in section 5, where the selection achieved by our method produces more precise estimations than the Manifold Parzen Windows and other approaches. Finally, Section 6 is devoted to conclusions.

2 The Smooth Parzen Windows Method

Let $\mathbf{x}$ be an D-dimensional random variable and $p()$ an arbitrary probability density function over $\mathbf{x}$ which is unknown and we want to estimate. The training set of the algorithm is formed by N samples of the random variable. For each training sample $\mathbf{x}_i$ we build a hard Q-neighbourhood H_i with the Q nearest neighbours of $\mathbf{x}_i$, including itself. Hence H_i is interpreted as a random event which happens iff the input belongs to that neighbourhood. The knowledge about the local structure of the distribution around $\mathbf{x}_i$ is obtained when we calculate the mean vector $\boldsymbol{\mu}$ and the correlation matrix $\mathbf{R}$:

$$\boldsymbol{\mu}(H_i) = E[\mathbf{x} \mid H_i] = \frac{1}{Q} \sum_{\mathbf{x}_j \in H_i} \mathbf{x}_j \tag{1}$$

$$\mathbf{R}(H_i) = E[\mathbf{x}\mathbf{x}^T \mid H_i] = \frac{1}{Q} \sum_{\mathbf{x}_j \in H_i} \mathbf{x}_j \mathbf{x}_j{}^T \tag{2}$$

Now we present a smoothing procedure to merge the information from different hard neighbourhoods. We define a soft cluster i by a random event S_i which verifies when the input belongs to cluster i. Each hard neighbourhood H_j contributes to S_i with a weight w_{ij}:

$$w_{ij} = P[H_j \mid S_i] \tag{3}$$

So, we have

$$\forall i \in \{1,2,...,M\}, \quad \sum_{j=1}^{N} w_{ij} = 1 \tag{4}$$

where the number of soft clusters M may be different from the number of hard neighbourhoods N. We can infer the structure of the soft cluster by merging the information from the hard neighbourhoods:

$$\boldsymbol{\mu}(S_i) = E[\mathbf{x} \mid S_i] = \sum_j P[H_j \mid S_i] E[\mathbf{x} \mid H_j] = \sum_{j=1}^{N} w_{ij} \boldsymbol{\mu}(H_j) \tag{5}$$

$$\mathbf{R}(S_i) = E[\mathbf{x}\mathbf{x}^T \mid S_i] = \sum_j P[H_j \mid S_i] E[\mathbf{x}\mathbf{x}^T \mid H_j] = \sum_{j=1}^{N} w_{ij} \mathbf{R}(H_j) \tag{6}$$

In order to define a Gaussian distribution we need the estimation of the covariance matrix $\mathbf{C}$ for each soft cluster:

$$\mathbf{C}(S_i) = E[(\mathbf{x} - \boldsymbol{\mu}(S_i))(\mathbf{x} - \boldsymbol{\mu}(S_i))^T \mid S_i] = \mathbf{R}(S_i) - \boldsymbol{\mu}(S_i)\boldsymbol{\mu}(S_i)^T \tag{7}$$

Finally, we need a method to determine the merging weights w_{ij}. We propose two approaches:

a) If M=N, we can perform the smoothing by replacing the 'hard' model at the data sample $\mathbf{x}_i$ by a weighted average of its neighbours ranked by their distance to $\mathbf{x}_i$. Here the model at $\mathbf{x}_i$ has the maximum weight, and their neighbours $\mathbf{x}_j$ have a weight which is a decreasing function of the distance from $\mathbf{x}_i$ to $\mathbf{x}_j$:

$$\omega_{ij} = \exp\left(- \frac{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2}{\psi^2} \right) \tag{8}$$

$$w_{ij} = \frac{\omega_{ij}}{\sum_{k=1}^{N} \omega_{ik}} \tag{9}$$

where ψ is a parameter to control the width of the smoothing. Please note that ω_{ii}=1.

b) We may use the fuzzy c-means algorithm [11] to perform a soft clustering. This algorithm partitions the set of training data into M clusters so it minimizes the distance within the cluster. The objective function is:

$$J = \sum_{i=1}^{M} \sum_{j=1}^{N} {m_{ij}}^{\phi} {d_{ij}}^2 \tag{10}$$

where ϕ is the fuzzy exponent which determines the degree of fuzzyness, and d_{ij} is the distance between training sample $\mathbf{x}_j$ and the centroid of cluster i.

The degrees of membership of training sample j to soft cluster i are obtained as m_{ij}, which can be regarded as the probability of training sample j belonging to cluster i. In this approach the weights w_{ij} of the local models that we merge to yield the model of cluster i are computed as follows:

$$w_{ij} = \frac{m_{ij}}{\sum_{k=1}^{N} m_{ik}} \tag{11}$$

3 Dynamic Parameter Selection

Once we have the estimations of the mean vectors $\boldsymbol{\mu}(S_i)$ and covariance matrices $\mathbf{C}(S_i)$ for each soft cluster S_i, it is needed to obtain a Gaussian distribution from them which is both accurate and small sized.

First we incorporate the capacity of estimating the intrinsic dimensionality. A second level of automated adaptation to the data will be added by means of a parameter γ. This parameter will enable the method to select the right noise level.

3.1 The Explained Variance Method

The explained variance method considers a variable number, K_i, of eigenvalues and their corresponding eigenvectors to be kept which is computed independently for each cluster S_i. Through the training process the method ensures that a minimum amount of variance is conserved in order to satisfy the level of accuracy, $\alpha \in [0,1]$, chosen by the user.

The lost variance when no directions are kept, V_0, can be defined as:

$$V_0 = \sum_{p=1}^{D} \lambda_i^p \tag{12}$$

with λ_i^p the p-th eigenvalue of the covariance matrix $\mathbf{C}(S_i)$, which are supposed to be sorted in decreasing order, and D the dimension of the training samples.

In any other situation the discarded variance, V_Z, depends on the number, Z, of principal directions conserved:

$$V_Z = \sum_{p=Z+1}^{D} \lambda_i^p \tag{13}$$

Let $V_0 - V_Z$ be the amount of error eliminated when we conserve information about the Z principal directions. Then

$$K_i = \min\left\{Z \in \{0,1,...,D\} \mid V_0 - V_Z \geq \alpha \ V_0\right\} \tag{14}$$

Substitution of (**12**) and (**13**) into (**14**) yields

$$K_i = \min\left\{Z \in \{0,1,...,D\} \mid \sum_{p=1}^{D} \lambda_i^p - \sum_{p=Z+1}^{D} \lambda_i^p \geq \alpha \ \sum_{p=1}^{D} \lambda_i^p\right\} \tag{15}$$

By expressing the sum of variances as the trace of $\mathbf{C}(S_i)$ we can simplify (**15**):

$$K_i = \min\left\{ Z \in \{0,1,...,D\} \mid \sum_{p=1}^{Z} \lambda_i^p \geq \alpha \ \ trace(\mathbf{C}(S_i)) \right\} \tag{16}$$

3.2 The Qualitative Parameter γ

We propose to select the noise variance parameter σ^2 as follows:

$$\sigma^2(S_i) = \gamma \cdot \lambda_i^{K_i} \tag{17}$$

where $\gamma \in [0,1]$ and $\lambda_i^{K_i}$ is the last of the preserved eigenvalues of $\mathbf{C}(S_i)$, i.e. the smallest of the first K_i largest eigenvalues. The estimated noise variance $\sigma^2(S_i)$ is added to the first K_i largest eigenvalues to yield the estimated variances of the K_i principal directions, while the D–K_i trailing directions have estimated variances of $\sigma^2(S_i)$.

3.3 Smooth Parzen Windows with Qualitative Parameters

The proposed algorithm is designed to estimate an unknown density distribution *p()* which the N samples of the training dataset are generated from. The generated estimator will be formed by a mixture of M Gaussians, one for each soft cluster:

$$\hat{p}(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} N_i(\mathbf{x}) \tag{18}$$

$$N_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(a_i + b_i) \right) \tag{19}$$

$$a_i = D\log(2\pi) + (D - K_i)\log\left(\lambda_i^p + \sigma^2(S_i)\right) + \sum_{p=1}^{K_i} \log\left(\lambda_i^p + \sigma^2(S_i)\right) \tag{20}$$

$$b_i = \frac{1}{\sigma^2(S_i)} \left\| \mathbf{x} - \boldsymbol{\mu}(S_i) \right\|^2 + \sum_{p=1}^{K_i} \left(\frac{1}{\lambda_i^p} - \frac{1}{\sigma^2(S_i)} \right) \left\| \left(\mathbf{u}_i^p\right)^T \left(\mathbf{x} - \boldsymbol{\mu}(S_i)\right) \right\|^2 \tag{21}$$

where $\mathbf{u}_i^p$ is the eigenvector corresponding to the p-th largest eigenvalue of $\mathbf{C}(S_i)$.

3.4 Summary

The training algorithm can be summarized as follows:

1. For each training sample, compute the mean vector $\boldsymbol{\mu}(H_i)$ and correlation matrix $\mathbf{C}(H_i)$ of its hard neighbourhood H_i with equations (**1**) and (**2**).
2. Estimate the merging weights w_{ij} either by the distance method (**9**) or the fuzzy c-means algorithm (**11**).

3. Compute the mean vectors $\boldsymbol{\mu}(S_i)$ and covariance matrices $\mathbf{C}(S_i)$ of each soft cluster S_i following (**5**) and (**7**).
4. Extract the eigenvalues and eigenvectors from $\mathbf{C}(S_i)$ and estimate the dimensionality of the underlying manifold K_i, by means of (**16**).
5. Use (**17**) to calculate $\sigma^2(S_i)$, the noise variance for the discarded directions.
6. Store each local model, i. e., the first K_i eigenvectors and eigenvalues, the local noise level $\sigma^2(S_i)$ and the mean vector $\boldsymbol{\mu}(S_i)$.

4 Convergence Proof

In this section we prove that our estimator $\hat{p}(\ )$ converges to the true density function *p()* in the limit $N\to\infty$ and $M\to\infty$.

Lemma 1. Every local Gaussian $N_i(\mathbf{x})$ tends to the D-dimensional Dirac delta function $\delta(\mathbf{x}-\boldsymbol{\mu}(\mathrm{S}_i))$ as $N\to\infty$ and $M\to\infty$.

Proof. In the limit $N\to\infty$ and $M\to\infty$ the clusters S_i reduce their volume to zero. This means that $\sigma^2(S_i)\to 0$ and $\lambda_i^p \to 0$ for all i and p. Hence the Gaussians $N_i(\mathbf{x})$ are confined to a shrinking volume centered at $\boldsymbol{\mu}(\mathrm{S}_i)$, because the variances in each direction are $\lambda_i^p + \sigma^2(S_i)$ or $\sigma^2(S_i)$, but they continue to integrate to 1. So, we have that $N_i(\mathbf{x}) \to \delta(\mathbf{x}-\boldsymbol{\mu}(\mathrm{S}_i))$.

Theorem 1. The expected value of the proposed estimation tends to the true probability density function as $N\to\infty$ and $M\to\infty$.

Proof. The expectation is w.r.t. the underlying distribution of the training samples, which is the true probability density function $p()$:

$$E[\hat{p}(\mathbf{x})] = \frac{1}{M}\sum_{i=1}^{M} E[N_i(\mathbf{x})] \tag{22}$$

Since $N_i(\mathbf{x})$ are independent and identically distributed random variables we get

$$E[\hat{p}(\mathbf{x})] = E[N_i(\mathbf{x})] = \int p(\mathbf{y}) N_{\mathbf{y}}(\mathbf{x}) d\mathbf{y} \tag{23}$$

where $N_{\mathbf{y}}()$ is a Gaussian centered in $\mathbf{y}$. Then, by Lemma 1, if $N\to\infty$ and $M\to\infty$ then $N_{\mathbf{y}}()$ shrinks to a Dirac delta:

$$E[\hat{p}(\mathbf{x})] \to \int p(\mathbf{y}) \delta(\mathbf{x}-\mathbf{y}) d\mathbf{y} \tag{24}$$

So, the expectation of the estimation converges to a convolution of the true density with the Dirac delta function. Then,

$$E[\hat{p}(\mathbf{x})] \to p(\mathbf{x}) \tag{25}$$

Theorem 2. The variance of the proposed estimation tends to zero as $N\to\infty$ and $M\to\infty$.

Proof. The variance is w.r.t. the underlying distribution of the training samples, which is the true probability density function $p()$:

$$\mathrm{var}[\hat{p}(\mathbf{x})] = \mathrm{var}\left[\frac{1}{M}\sum_{i=1}^{M} N_i(\mathbf{x})\right] \tag{26}$$

Since $N_i(\mathbf{x})$ are independent and identically distributed random variables we get

$$\mathrm{var}[\hat{p}(\mathbf{x})] = \frac{1}{M}\mathrm{var}[N_i(\mathbf{x})] \tag{27}$$

By the properties of variance and (**23**) we obtain

$$\mathrm{var}[\hat{p}(\mathbf{x})] = \frac{1}{M}\left(E\left[(N_i(\mathbf{x}))^2\right] - E[N_i(\mathbf{x})]^2\right) = \frac{1}{M}\left(E\left[(N_i(\mathbf{x}))^2\right] - E[\hat{p}(\mathbf{x})]^2\right) \tag{28}$$

By definition of expectation

$$\mathrm{var}[\hat{p}(\mathbf{x})] = \frac{1}{M}\left(\int p(\mathbf{y})(N_{\mathbf{y}}(\mathbf{x}))^2 d\mathbf{y} - E[\hat{p}(\mathbf{x})]^2\right) \tag{29}$$

where again $N_{\mathbf{y}}()$ is a Gaussian centered in $\mathbf{y}$. We can bound the integral of the above equation with the help of (**23**), and so we get

$$\mathrm{var}[\hat{p}(\mathbf{x})] \le \frac{\sup(N(\cdot))E[\hat{p}(\mathbf{x})]}{M} \to 0 \quad \text{as } N\to\infty \text{ and } M\to\infty \tag{30}$$

5 Experimental Results

This section shows some experiments we have designed in order to study the quality of the density estimation achieved by our method. We call it SmoothDist when the distance weighting is used, and SmoothFuzzy when we use fuzzy c-means. Vincent and Bengio's method is referred as MParzen, the original Parzen windows method (with isotropic Gaussian kernels) is called OParzen, and finally the Mixtures of Probabilistic PCA model of Tipping and Bishop [12] is called MPPCA. For this purpose the performance measure we have chosen is the average negative log likelihood

$$ANLL = -\frac{1}{T}\sum_{i=1}^{T}\log \hat{p}(\mathbf{x}_i) \tag{31}$$

where $\hat{p}()$ is the estimator, and the test dataset is formed by T samples $\mathbf{x}_i$.

5.1 Experiment on 2D Artificial Data

A training set of 100 points, a validation set of 100 points and a test set of 10000 points were generated from the following distribution of two dimensional *(x,y)* points:

$$x = 0.04t\sin(t) + \varepsilon_x, \quad y = 0.04t\cos(t) + \varepsilon_y \tag{32}$$

where $t \sim U(3,15), \varepsilon_x \sim N(0,0.01), \varepsilon_y \sim N(0,0.01), U(a,b)$ is uniform in the interval *(a,b)* and $N(\mu,\sigma)$ is a normal density.

We have optimized separately all the parameters of the five competing models with disjoint training and validation sets. The performance of the optimized models has been computed by 10-fold cross-validation, and the results are shown in Table 1, with the best result marked in bold. It can be seen that our models outperform the other three in density distribution estimation.

Table 1. Quantitative results on the espiral dataset (standard deviations in parentheses)

Method	Optimized parameters used	ANLL on test set
SmoothDist	M=100, α=0.9, Q=4, γ=0.03, ψ=0.001	–1.5936 (0.2557)
SmoothFuzzy	M=100, α=0.1, Q=4, γ=0.05	**–1.6073 (0.3293)**
OParzen	M=100, σ^2=0.0001	1.0817 (1.3357)
MParzen	M=100, K=2, Q=4, σ^2=1.6E–5	–0.9505 (0.3301)
MPPCA	M=4, K=1	0.2473 (0.0818)

Fig. 1. Density estimation for the 2D artificial dataset. From left to right and from top to bottom: SmoothDist, SmoothFuzzy, OParzen, MParzen and MPPCA.

Figure 1 shows density distribution plots corresponding to the five models. Darker areas represent zones with high density mass and lighter ones indicate the estimator has detected a low density area.

We can see in the plots that our models have less density holes (light areas) and less ‘bumpiness’. This means that our model represents more accurately the true distribution, which has no holes and is completely smooth. We can see that the quantitative ANLL results agree with the plots, because the lowest values of ANLL

match the best-looking plots. So, our model outperforms clearly the other three considered approaches.

5.2 Density Estimation Experiment

A density estimation experiment has been designed, where we have chosen three multidimensional datasets from the UCI Repository of Machine Learning Databases [13]. As in the previous experiment, we have optimized all the parameters of the five competing models with disjoint training and validation sets. The parameters for the density estimator of each dataset have been optimized separately. Table 2 shows the results of the 10-fold cross-validation, with the winning models in bold. Our two proposals show a superior performance.

Table 2. ANLL on test set (standard deviations in parentheses)

Dataset	SmoothDist	SmoothFuzzy	OParzen	MParzen	MPPCA
auto-mpg	**25.4 (0.8)**	26.3 (2.4)	29.3 (3.8)	30.1 (3.7)	33.1 (1.7)
cloud	24.9 (0.7)	**24.8 (0.6)**	36.9 (3.5)	37.7 (3.5)	31.6 (1.0)
housing	25.7 (1.2)	**24.9 (1.7)**	31.2 (2.2)	32.1 (2.2)	43.4 (4.9)

5.3 Classification Experiment

We have selected three classification benchmarks from the UCI Repository of Machine Learning Databases [13] to perform a classification experiment. We have considered Bayesian classifiers which are built by estimating the probability density function of each class separately. Then, a test pattern is assigned to the class which yields the largest probability density. We have optimized the model parameters separately, as in the previous experiment. In this case we have optimized the models independently for each class of each database. The results of the 10-fold cross-validation are shown in Table 3, and the winning model for each database is shown in bold. We can see that our two approaches outperform the other three.

Table 3. Successful classification percentages on test set (standard deviations in parentheses)

Database	SmoothDist	SmoothFuzzy	OParzen	MParzen	MPPCA
glass	**69.2 (11.5)**	66.7 (8.5)	63.0 (14.6)	64.0 (10.4)	46.4 (11.2)
liver	67.4 (5.9)	**68.8 (7.2)**	60.6 (7.9)	62.6 (7.8)	59.2 (10.9)
pima	**69.9 (5.0)**	61.6 (5.4)	66.0 (3.9)	62.0 (4.9)	62.9 (5.6)

6 Conclusions

We have presented a probability density estimation model. It is based in the Parzen window approach. Our proposal builds local models for a hard neighbourhood of each training sample. Then soft clusters are obtained by merging these local models. A local Gaussian density is developed by selecting independently for each soft cluster the best number of retained dimensions and the best estimation of noise variance. This

allows our method to represent input distributions more faithfully than the Manifold Parzen window model, which is an improvement of the original Parzen window method. Computational results show the superior performance of our method.

Acknowledgements

This work was partially supported by the Ministry of Education and Science of Spain under Projects TIN2005-02984 and TIN2006-07362.

References

1. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
2. Silverman, B.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, New York (1986)
3. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
4. Izenman, A.J.: Recent developments in nonparametric density estimation. Journal of the American Statistical Association 86(413), 205–224 (1991)
5. Lejeune, M., Sarda, P.: Smooth estimators of distribution and density functions. Computational Statistics & Data Analysis 14, 457–471 (1992)
6. Hjort, N.L., Jones, M.C.: Locally Parametric Nonparametric Density Estimation. Annals of Statistics 24(4), 1619–1647 (1996)
7. Hastie, T., Loader, C.: Local regression: Automatic kernel carpentry. Statistical Science 8, 120–143 (1993)
8. McLachlan, G., Peel, D.: Finite Mixture Models. Wiley, Chichester (2000)
9. Parzen, E.: On the Estimation of a Probability Density Function and Mode. Annals of Mathematical Statistics 33, 1065–1076 (1962)
10. Vincent, P., Bengio, Y.: Manifold Parzen Windows. Advances in Neural Information Processing Systems 15, 825–832 (2003)
11. Bezdek, J.C.: Numerical taxonomy with fuzzysets. J. Math. Biol. 1, 57–71 (1974)
12. Tipping, M.E., Bishop, C.M.: Mixtures of Probabilistic Principal Components Analyzers. Neural Computation 11, 443–482 (1999)
13. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. University of California, Department of Information and Computer Science, Irvine, CA (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html